**[html]** [view plain](http://blog.csdn.net/pltuyuhong/article/details/46974039) [copy](http://blog.csdn.net/pltuyuhong/article/details/46974039)

[print?](http://blog.csdn.net/pltuyuhong/article/details/46974039)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans**
3. xmlns="http://www.springframework.org/schema/beans"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xmlns:p="http://www.springframework.org/schema/p"
6. xmlns:context="http://www.springframework.org/schema/context"
7. xsi:schemaLocation=
8. "http://www.springframework.org/schema/beans
9. http://www.springframework.org/schema/beans/spring-beans.xsd
10. http://www.springframework.org/schema/context
11. http://www.springframework.org/schema/context/spring-context-3.0.xsd"**>**
13. <!-- 自动扫描bean -->
14. **<context:component-scan** base-package="com.sq.platform.testData"**/>**
16. **<bean** id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"**>**
17. **<property** name="locations"**>**
18. **<list>**
19. **<value>**classpath:conf/conf.properties**</value>**
20. **</list>**
21. **</property>**
22. **</bean>**
23. **<bean** id="dataSourceOne" class="org.apache.commons.dbcp.BasicDataSource"
24. destroy-method="close" p:driverClassName="${drivenClassName}" p:url="${jdbcUrlOne}"
25. p:username="${user}" p:password="${password}" **/>**
27. **<bean** id="dataSourceTwo" class="org.apache.commons.dbcp.BasicDataSource"
28. destroy-method="close" p:driverClassName="${drivenClassName}" p:url="${jdbcUrlTwo}"
29. p:username="${user}" p:password="${password}" **/>**
31. **<bean** id="dynamicDataSource" class="com.sq.platform.testData.dao.DynamicDataSource" **>**
32. <!-- 通过key-value的形式来关联数据源 -->
33. **<property** name="targetDataSources"**>**
34. **<map>**
35. **<entry** value-ref="dataSourceOne" key="one"**></entry>**
36. **<entry** value-ref="dataSourceTwo" key="Two"**></entry>**
37. **</map>**
38. **</property>**
39. **<property** name="defaultTargetDataSource" ref="dataSourceOne" **/>**
40. **</bean>**
42. <!--JdbcTemplate使用动态数据源的配置-->
43. **<bean** id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate"**>**
44. **<property** name="dataSource"**>**
45. **<ref** bean="dynamicDataSource" **/>**
46. **</property>**
47. **</bean>**
48. **</beans>**

2.资源配置文件conf.properties

**[html]** [view plain](http://blog.csdn.net/pltuyuhong/article/details/46974039) [copy](http://blog.csdn.net/pltuyuhong/article/details/46974039)

[print?](http://blog.csdn.net/pltuyuhong/article/details/46974039)

1. #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*database configure\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. drivenClassName=org.postgresql.Driver
3. jdbcUrlOne=jdbc:xxxx
4. jdbcUrlTwo=jdbc:xxxxx
5. user=admin
6. password=admin

3.创建两个类DBContextHolder.[**Java**](http://lib.csdn.net/base/javase) 和 DynamicDataSource.java

**[html]** [view plain](http://blog.csdn.net/pltuyuhong/article/details/46974039) [copy](http://blog.csdn.net/pltuyuhong/article/details/46974039)

[print?](http://blog.csdn.net/pltuyuhong/article/details/46974039)

1. package com.sq.platform.testData.dao;

4. public class DBContextHolder{
5. public static final String OneUrl= "one";
6. public static final String TwoUrl= "two";
8. private static final ThreadLocal**<String>** contextHolder = new ThreadLocal**<String>**();
10. public static void setDBType(String dbType) {
11. contextHolder.set(dbType);
12. }
14. public static String getDBType() {
15. return contextHolder.get();
16. }
18. public static void clearDBType() {
19. contextHolder.remove();
20. }
21. }

**[html]** [view plain](http://blog.csdn.net/pltuyuhong/article/details/46974039) [copy](http://blog.csdn.net/pltuyuhong/article/details/46974039)

[print?](http://blog.csdn.net/pltuyuhong/article/details/46974039)

1. package com.sq.platform.testData.dao;
3. import org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;
5. public class DynamicDataSource extends AbstractRoutingDataSource {
7. @Override
8. protected Object determineCurrentLookupKey() {
9. return DBContextHolder.getDBType();
10. }
11. }

4.调用方法，每次调用jdbc之前需要设置数据源

**[html]** [view plain](http://blog.csdn.net/pltuyuhong/article/details/46974039) [copy](http://blog.csdn.net/pltuyuhong/article/details/46974039)

[print?](http://blog.csdn.net/pltuyuhong/article/details/46974039)

1. @Autowired
2. private JdbcTemplate jdbcTemplate;
4. public void insertData(String tableName, int value\_id, float value, String sampleTime){
5. String sql = "insert into " + tableName+ " values(?,?,?)";
6. DBContextHolder.setDBType(DBContextHolder.OneUrl);
7. jdbcTemplate.update(sql, new Object[]{value\_id, value, sampleTime}, new int[]{Types.INTEGER, Types.FLOAT, Types.TIMESTAMP});
8. }

[http://static.blog.csdn.net/images/save_snippets.png](javascript:;)